**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ’ЯЗКУ**

**ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ**

**«ФАХОВИЙ КОЛЕДЖ ЗВ’ЯЗКУ ТА ІНФОРМАТИЗАЦІЇ**

**ДЕРЖАВНОГО УНІВЕРСИТЕТУ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ’ЯЗКУ»**

Циклова комісія інформаційних технологій

**Звіт з практичної роботи №3**

**З дисципліни: об'єктно-орієнтоване програмування**

Виконав: студенти 3 курсу, групи КН-31 спеціальності

122 Комп’ютерні науки

освітня програма

Обслуговування програмних систем і комплексів

Ракітенко Артем

Керівник Андрій Куляк

Одеса – 2025

**ЗМІСТ**

[1.МЕТА РОБОТИ 4](#_Toc210737624)

[2.ЗАВДАННЯ 4](#_Toc210737625)

[3.ДІАГРАМА КЛАСІВ 5](#_Toc210737626)

[4.СТРУКТУРА ПРОГРАМИ 6](#_Toc210737627)

[5.КОД ПРОГРАМИ 7](#_Toc210737628)

[5.1 User.java (фрагмент з інкапсуляцією): 7](#_Toc210737629)

[5.2 BankingService.java (фрагмент): 8](#_Toc210737630)

[6.РЕЗУЛЬТАТИ ВИКОНАННЯ 9](#_Toc210737631)

[7.ВИСНОВКИ 10](#_Toc210737632)

# **МЕТА РОБОТИ**

Опанувати практичні навички застосування принципів інкапсуляції в мові програмування Java. Навчитися використовувати різні специфікатори доступу, створювати бізнес-методи замість простих сетерів, організовувати код у пакети для демонстрації package-private методів.

# **ЗАВДАННЯ**

1. Переробити діаграму класів з практичної роботи №2 з урахуванням принципів інкапсуляції

2. Зробити всі поля класів непублічними та реалізувати методи доступу

3. Використовувати різні специфікатори доступу (public, protected, package-private, private)

4. Створити бізнес-методи замість простих геттерів/сеттерів

5. Організувати код у пакети для демонстрації package-private методів

6. Реалізувати функціонал додавання та управління сутностями

# **ДІАГРАМА КЛАСІВ**

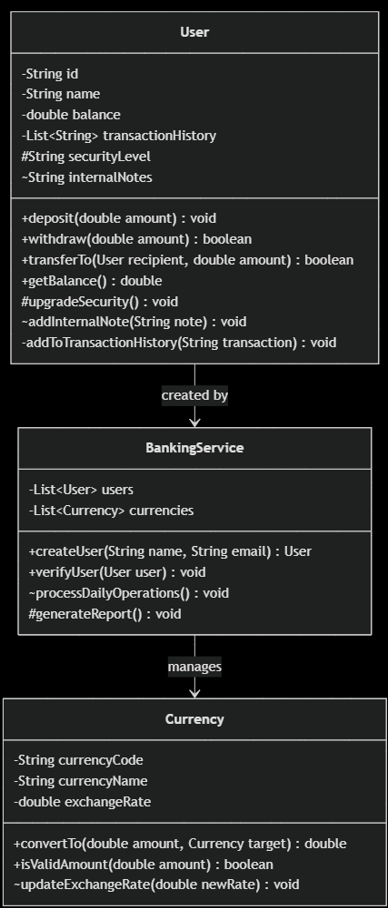


Рисунок 3.1 – Діаграма класів

# **4. СТРУКТУРА ПРОГРАМИ**

Програма має наступну структуру класів:

src/

└── ua/

└── edu/

└── ontu/

└── cryptobanking/

├── Main.java

├── model/

│ ├── Currency.java

│ ├── Deposit.java

│ ├── Exchange.java

│ ├── Operation.java

│ ├── Transfer.java

│ ├── User.java

│ └── Withdrawal.java

└── service/

└── BankingService.java

# **КОД ПРОГРАМИ**

## **5.1 User.java (фрагмент з інкапсуляцією):**

public class User {

private String id;

private String name;

private double balance;

private List<String> transactionHistory;

String internalNotes;

protected String securityLevel;

public void deposit(double amount) {

if (amount > 0) {

this.balance += amount;

addToTransactionHistory("DEPOSIT: +" + amount);

}

}

private void addToTransactionHistory(String transaction) {

transactionHistory.add(transaction);

}

void addInternalNote(String note) {

this.internalNotes += note + "\n";

}

}

## **5.2 BankingService.java (фрагмент):**

public class BankingService {

private List<User> users;

public User createUser(String name, String email) { // public метод

User user = new User(userId, name);

user.setContactInfo(email);

users.add(user);

return user;

}

void processDailyOperations() { // package-private метод

// внутрішня логіка

}

protected void generateReport() { // protected метод

System.out.println("Bank Report: " + users.size() + " users");

}

}

# **6. РЕЗУЛЬТАТИ ВИКОНАННЯ**

При запуску програми отримано наступний результат:

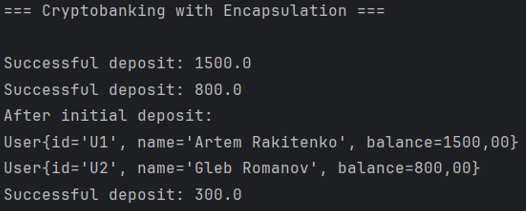


Рисунок 6.1 – Початкове створення користувачів та поповнення

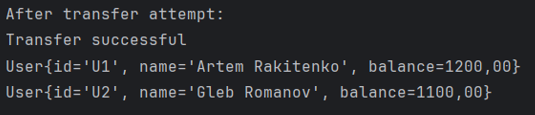


Рисунок 6.2 – Демонстрація переказу коштів

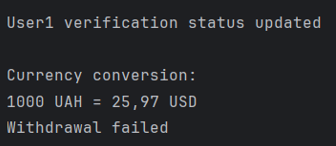


Рисунок 6.3 – Історія транзакцій

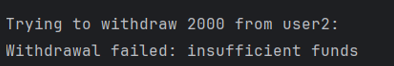


Рисунок 6.4 – Демонстрація валідації та помилок

# **7. ВИСНОВКИ**

У ході виконання практичної роботи №3 було успішно застосовано принципи інкапсуляції

у системі криптобанкінгу. Основні результати:

1. Реалізовано повну інкапсуляцію даних - всі поля класів оголошено як private

2. Створено бізнес-методи (deposit, withdraw, transferTo) замість простих сетерів

3. Використано різні специфікатори доступу: public, protected, package-private, private

4. Організовано структуру пакетів для демонстрації package-private методів

5. Реалізовано функціонал управління користувачами та операціями